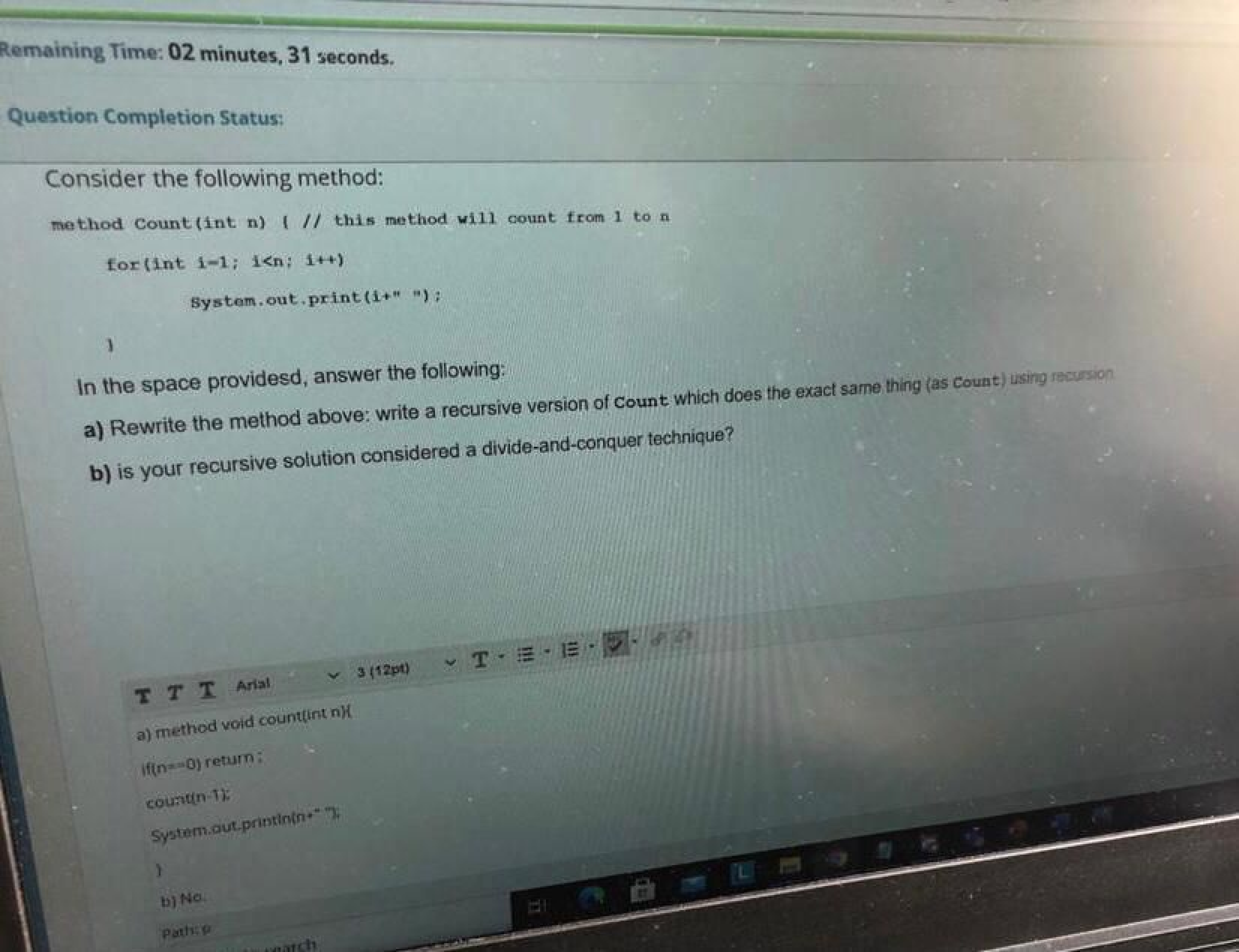


Remaining Time: 02 minutes, 31 seconds.

Question Completion Status:

Consider the following method:

```
method Count(int n) { // this method will count from 1 to n

      for(int i=1; i<n; i++)

            System.out.print(i+" ");

   }
```

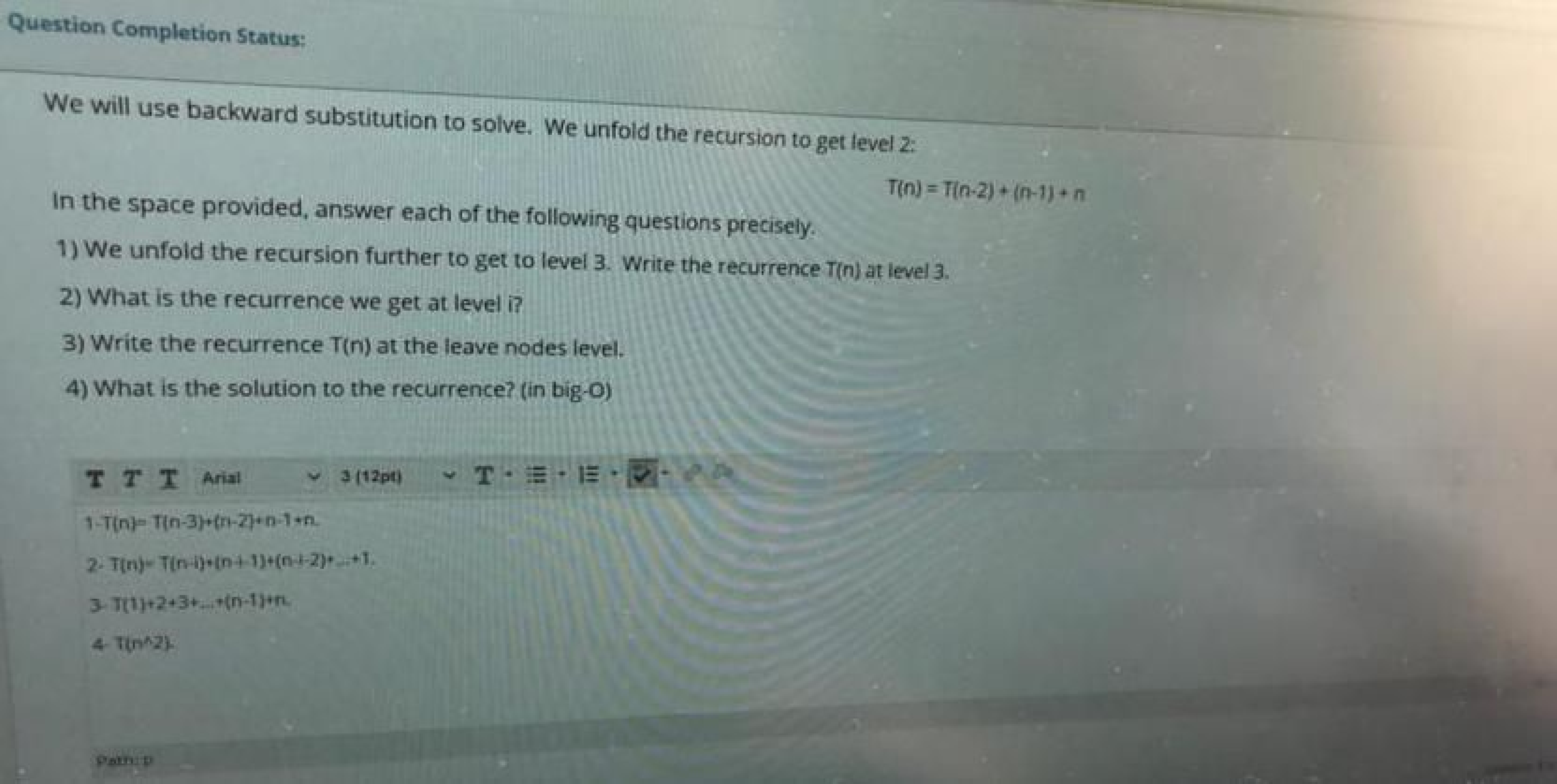
In the space providesd, answer the following:

a) Rewrite the method above: write a recursive version of `Count` which does the exact same thing (as `Count`) using recursion

b) is your recursive solution considered a divide-and-conquer technique?

```
a) method void count(int n){
if(n==0) return ;
count(n-1);
System.out.println(n+" ");
}
```

b) No.



Question Completion Status:

We will use backward substitution to solve. We unfold the recursion to get level 2:

T(n) = T(n-2) + (n-1) + n

In the space provided, answer each of the following questions precisely.

1) We unfold the recursion further to get to level 3. Write the recurrence T(n) at level 3.

2) What is the recurrence we get at level i?

3) Write the recurrence T(n) at the leave nodes level.

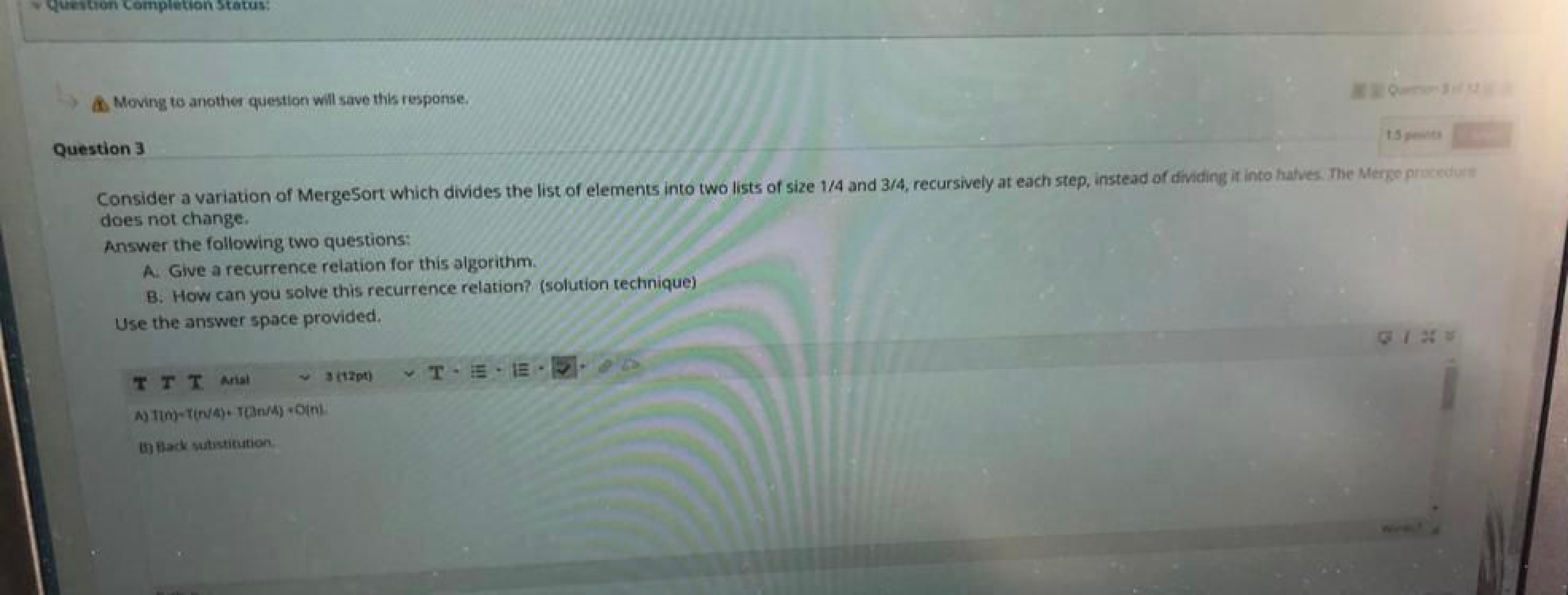
4) What is the solution to the recurrence? (in big-O)

1-T(n)= T(n-3)+(n-2)+n-1+n.

2- T(n)= T(n-i)+(n-i-1)+(n-i-2)+...+1.

3- T(1)+2+3+...+(n-1)+n.

4- T(n^2).



Question Completion Status:

Moving to another question will save this response.

Question 3 of 12

Question 3

15 points

Consider a variation of MergeSort which divides the list of elements into two lists of size 1/4 and 3/4, recursively at each step, instead of dividing it into halves. The Merge procedure does not change.

Answer the following two questions:

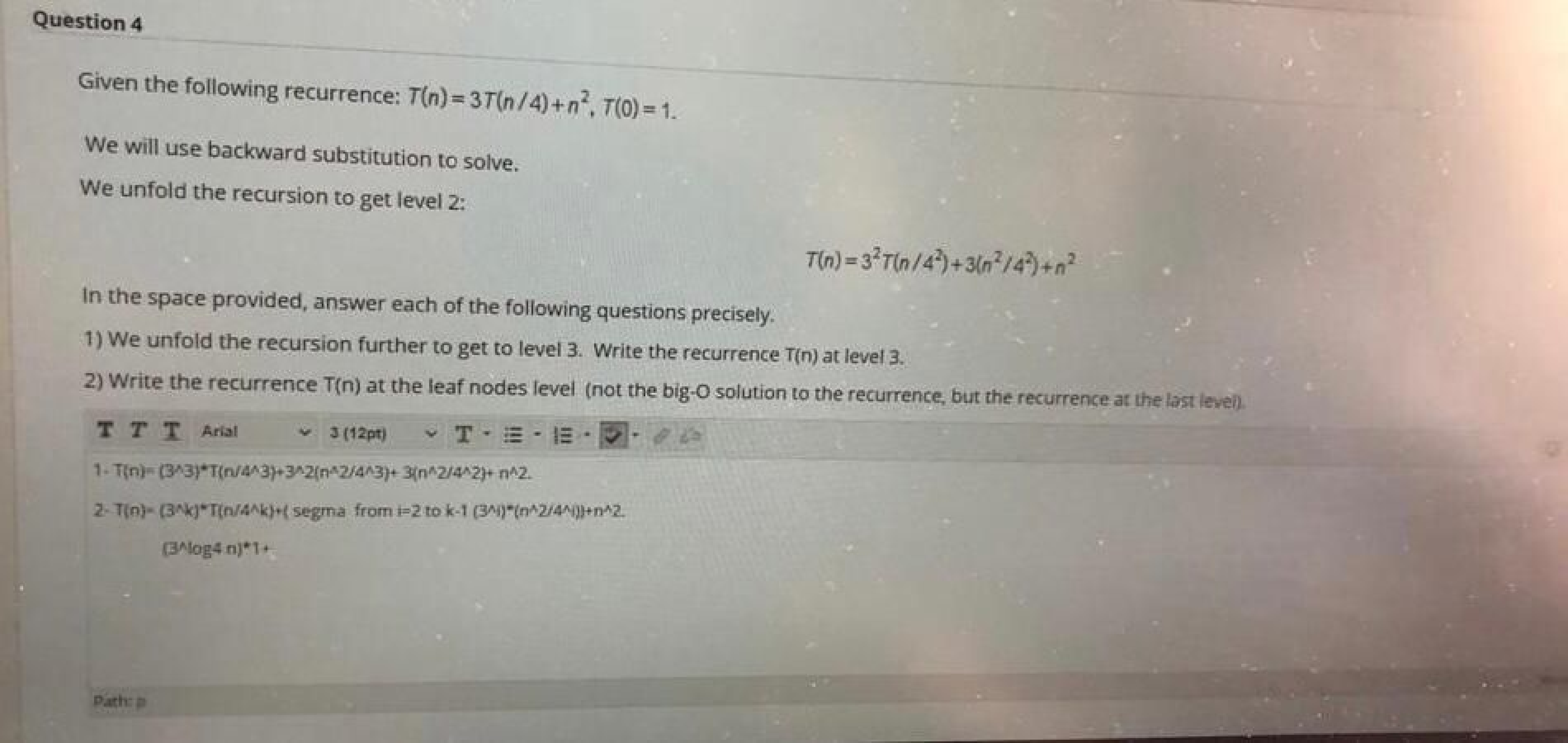
A. Give a recurrence relation for this algorithm.

B. How can you solve this recurrence relation? (solution technique)

Use the answer space provided.

A) $T(n)=T(n/4)+T(3n/4)+O(n)$.

B) Back substitution.



Question 4

Given the following recurrence: $T(n) = 3T(n/4) + n^2$, $T(0) = 1$.

We will use backward substitution to solve.

We unfold the recursion to get level 2:

$$T(n) = 3^2 T(n/4^2) + 3(n^2/4^2) + n^2$$

In the space provided, answer each of the following questions precisely.

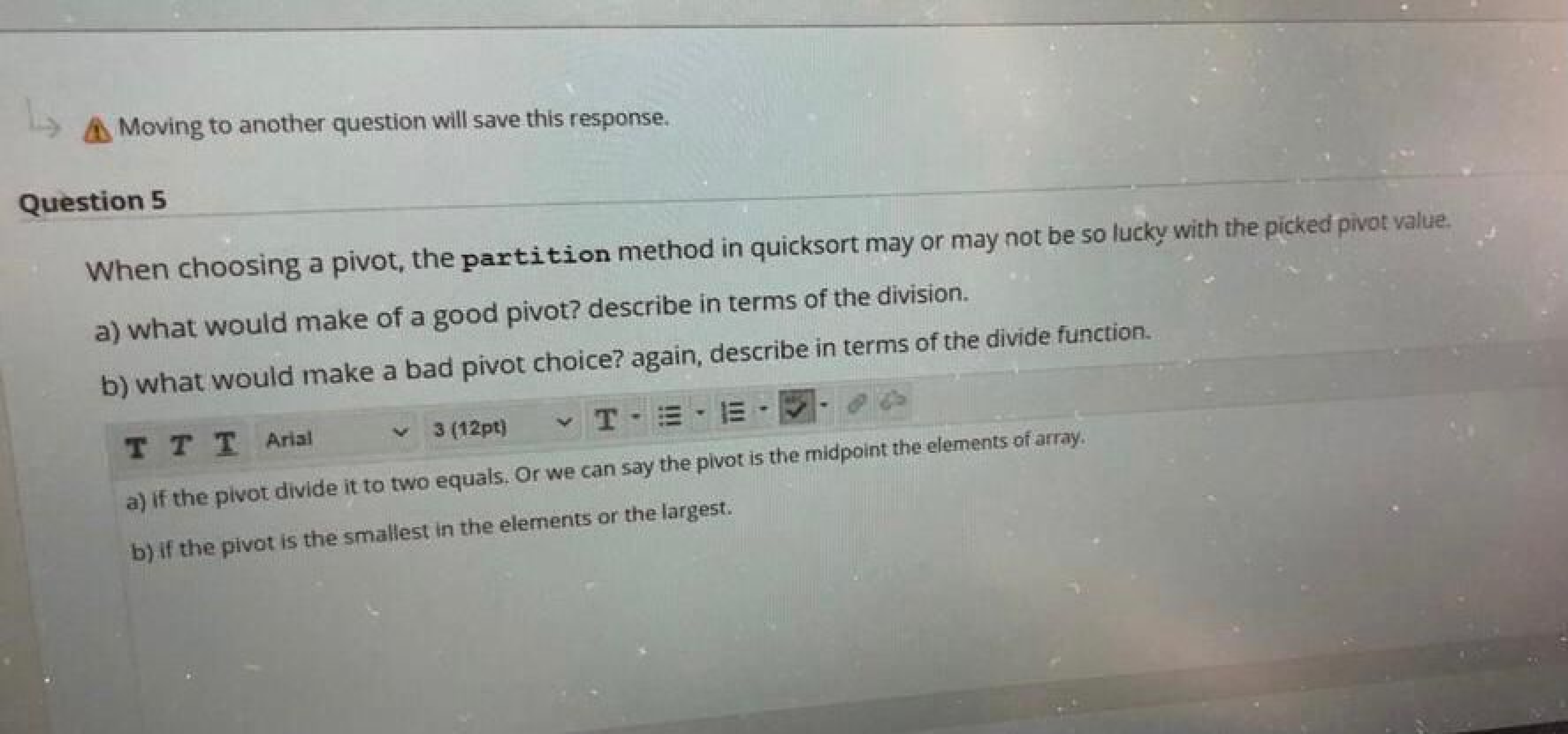
1) We unfold the recursion further to get to level 3. Write the recurrence T(n) at level 3.

2) Write the recurrence T(n) at the leaf nodes level (not the big-O solution to the recurrence, but the recurrence at the last level).

1- T(n)= (3^3)*T(n/4^3)+3^2(n^2/4^3)+ 3(n^2/4^2)+ n^2.

2- T(n)= (3^k)*T(n/4^k)+(segma from i=2 to k-1 (3^i)*(n^2/4^i))+n^2.

(3^log4 n)*1+



⚠ Moving to another question will save this response.

Question 5

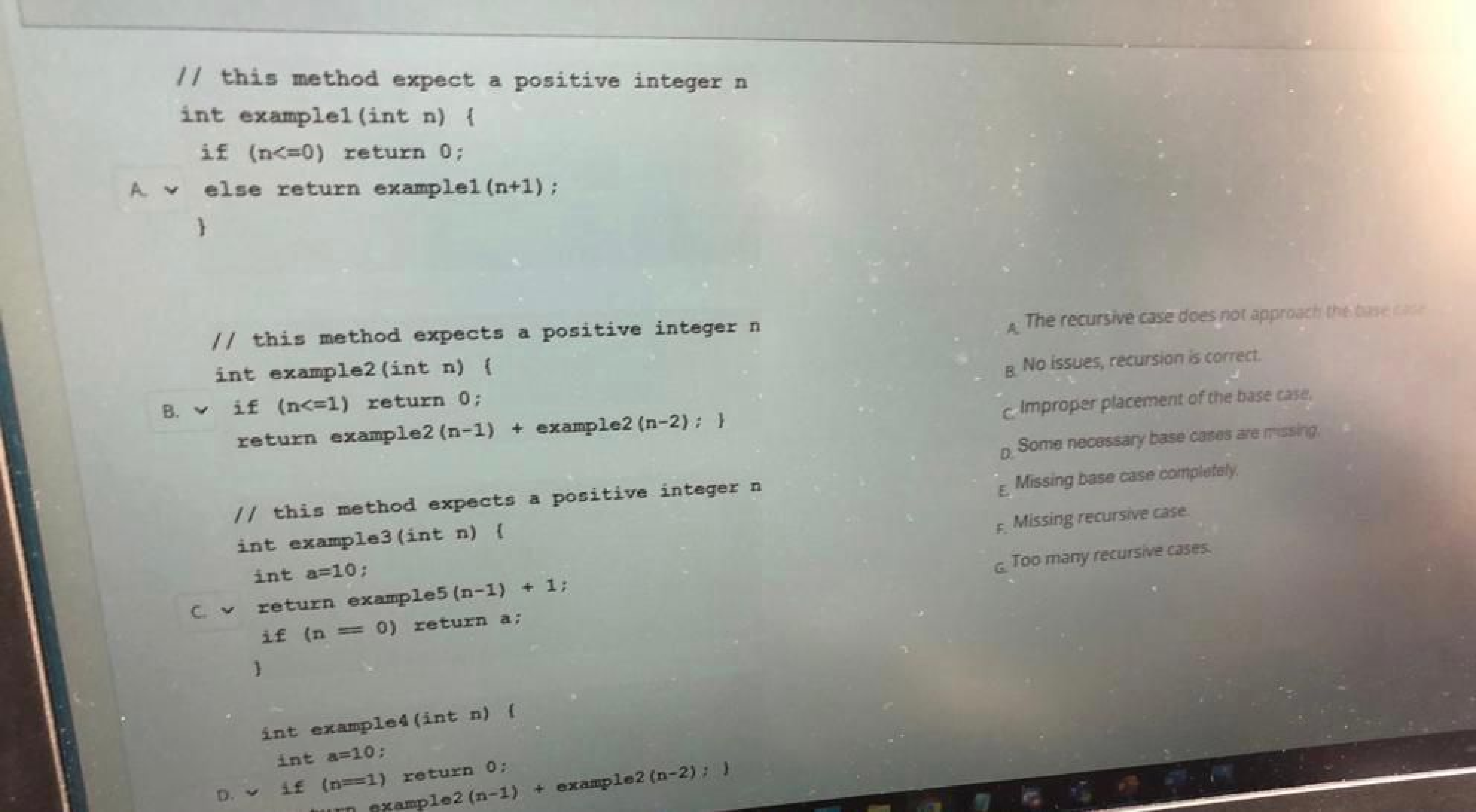
When choosing a pivot, the `partition` method in quicksort may or may not be so lucky with the picked pivot value.

a) what would make of a good pivot? describe in terms of the division.

b) what would make a bad pivot choice? again, describe in terms of the divide function.

a) if the pivot divide it to two equals. Or we can say the pivot is the midpoint the elements of array.

b) if the pivot is the smallest in the elements or the largest.



```
// this method expect a positive integer n
int example1(int n) {
  if (n<=0) return 0;
A. ⌄  else return example1(n+1);
  }
```

```
// this method expects a positive integer n
int example2(int n) {
B. ⌄  if (n<=1) return 0;
  return example2(n-1) + example2(n-2); }
```

```
// this method expects a positive integer n
int example3(int n) {
  int a=10;
C. ⌄  return example5(n-1) + 1;
  if (n == 0) return a;
  }
```

```
int example4(int n) {
  int a=10;
D. ⌄  if (n==1) return 0;
  urn example2(n-1) + example2(n-2); }
```

A. The recursive case does not approach the base case

B. No issues, recursion is correct.

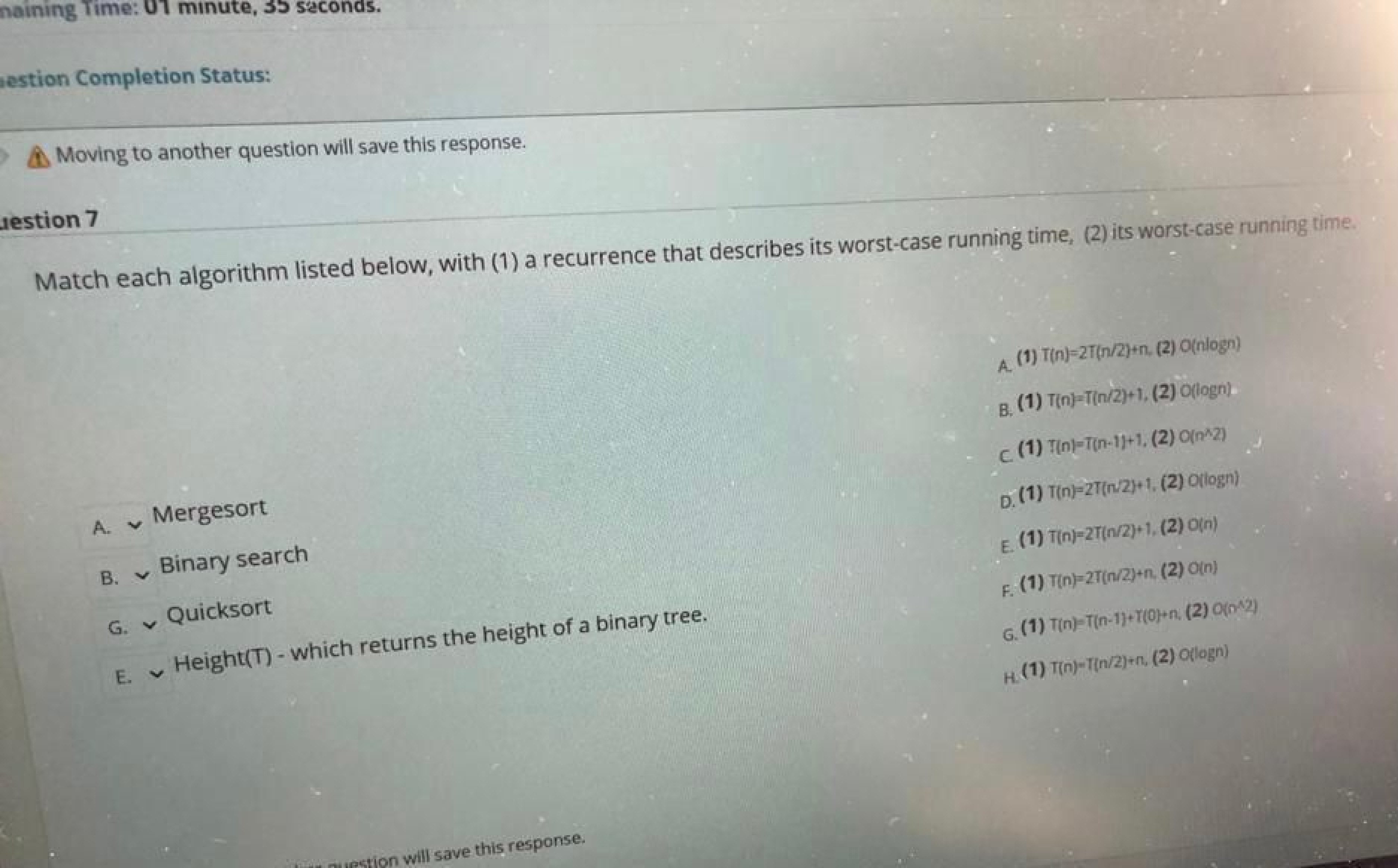
C. Improper placement of the base case.

D. Some necessary base cases are missing.

E. Missing base case completely.

F. Missing recursive case.

G. Too many recursive cases.



naining Time: 01 minute, 35 seconds.

estion Completion Status:

⚠ Moving to another question will save this response.

uestion 7

Match each algorithm listed below, with (1) a recurrence that describes its worst-case running time, (2) its worst-case running time.

A. ⌄ Mergesort

B. ⌄ Binary search

G. ⌄ Quicksort

E. ⌄ Height(T) - which returns the height of a binary tree.

A. **(1)** T(n)=2T(n/2)+n, **(2)** O(nlogn)

B. **(1)** T(n)=T(n/2)+1, **(2)** O(logn)

C. **(1)** T(n)=T(n-1)+1, **(2)** O(n^2)

D. **(1)** T(n)=2T(n/2)+1, **(2)** O(logn)

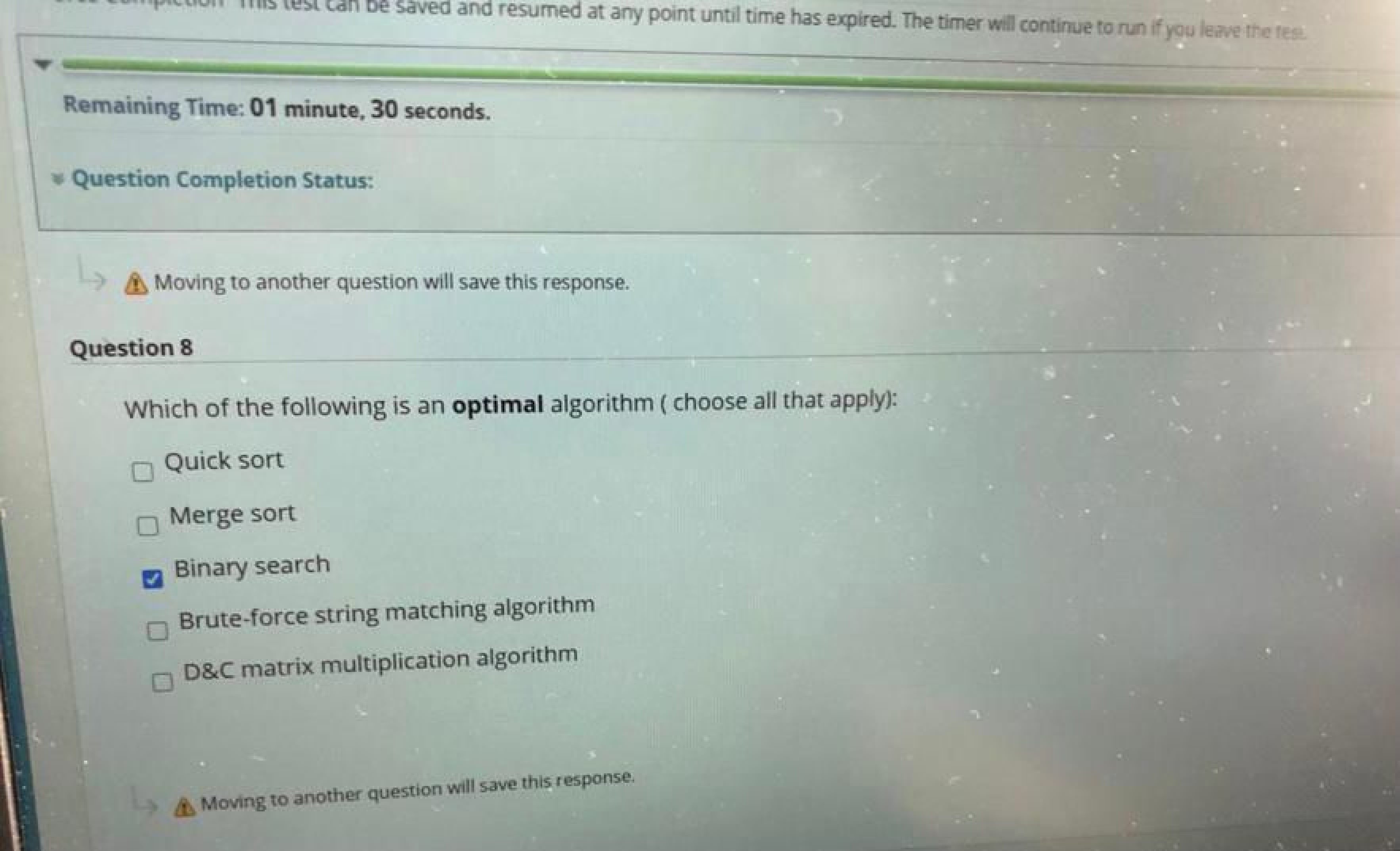
E. **(1)** T(n)=2T(n/2)+1, **(2)** O(n)

F. **(1)** T(n)=2T(n/2)+n, **(2)** O(n)

G. **(1)** T(n)=T(n-1)+T(0)+n, **(2)** O(n^2)

H. **(1)** T(n)=T(n/2)+n, **(2)** O(logn)

question will save this response.



This test can be saved and resumed at any point until time has expired. The timer will continue to run if you leave the test.

Remaining Time: **01 minute, 30 seconds.**

Question Completion Status:

⚠ Moving to another question will save this response.

Question 8

Which of the following is an **optimal** algorithm (choose all that apply):

☐ Quick sort

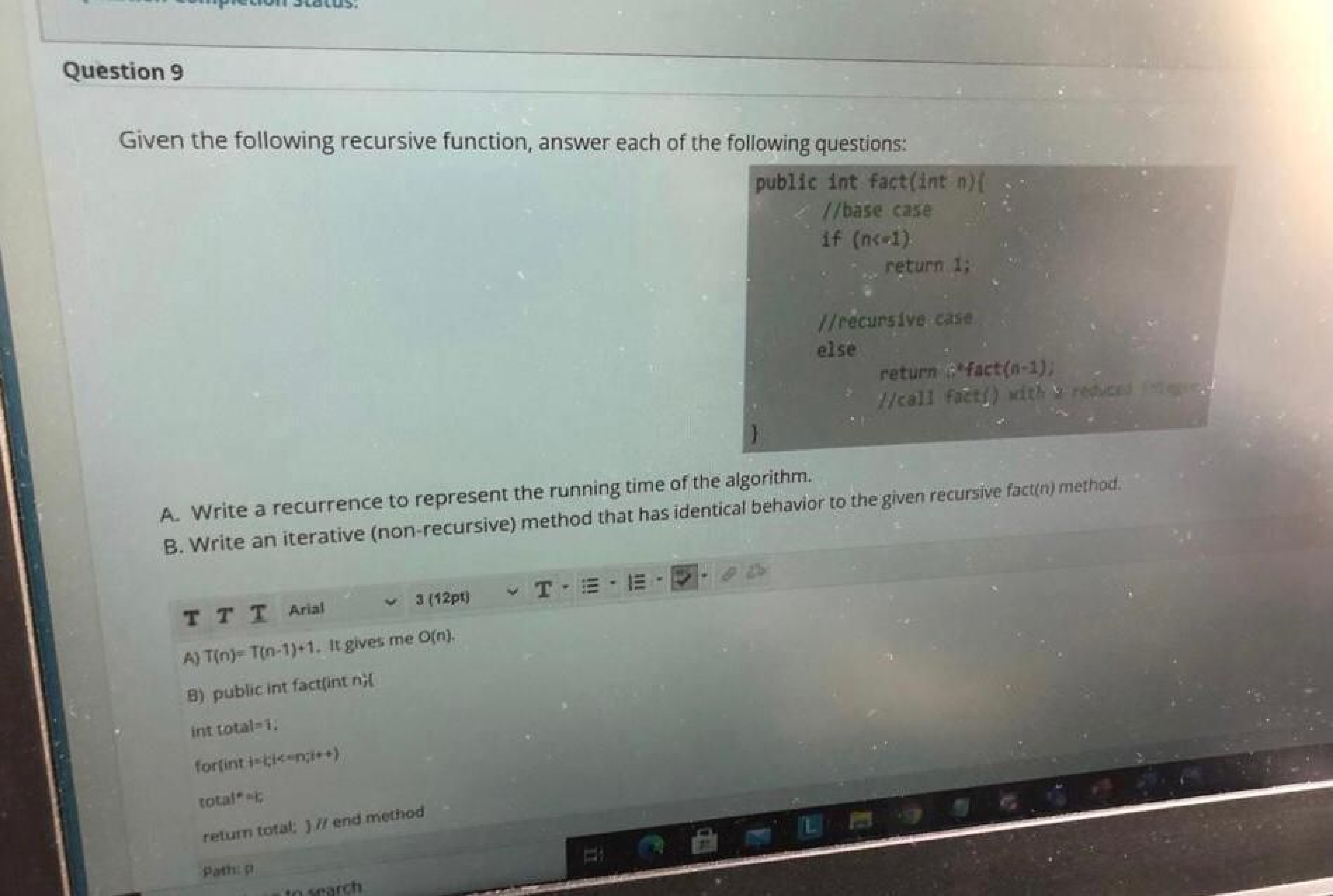
☐ Merge sort

☑ Binary search

☐ Brute-force string matching algorithm

☐ D&C matrix multiplication algorithm

⚠ Moving to another question will save this response.



Question 9

Given the following recursive function, answer each of the following questions:

```
public int fact(int n){
      //base case
      if (n<=1)
            return 1;

      //recursive case
      else
            return n*fact(n-1);
            //call fact() with a reduced integer
}
```

A. Write a recurrence to represent the running time of the algorithm.

B. Write an iterative (non-recursive) method that has identical behavior to the given recursive *fact(n)* method.

A) T(n)= T(n-1)+1. It gives me O(n).

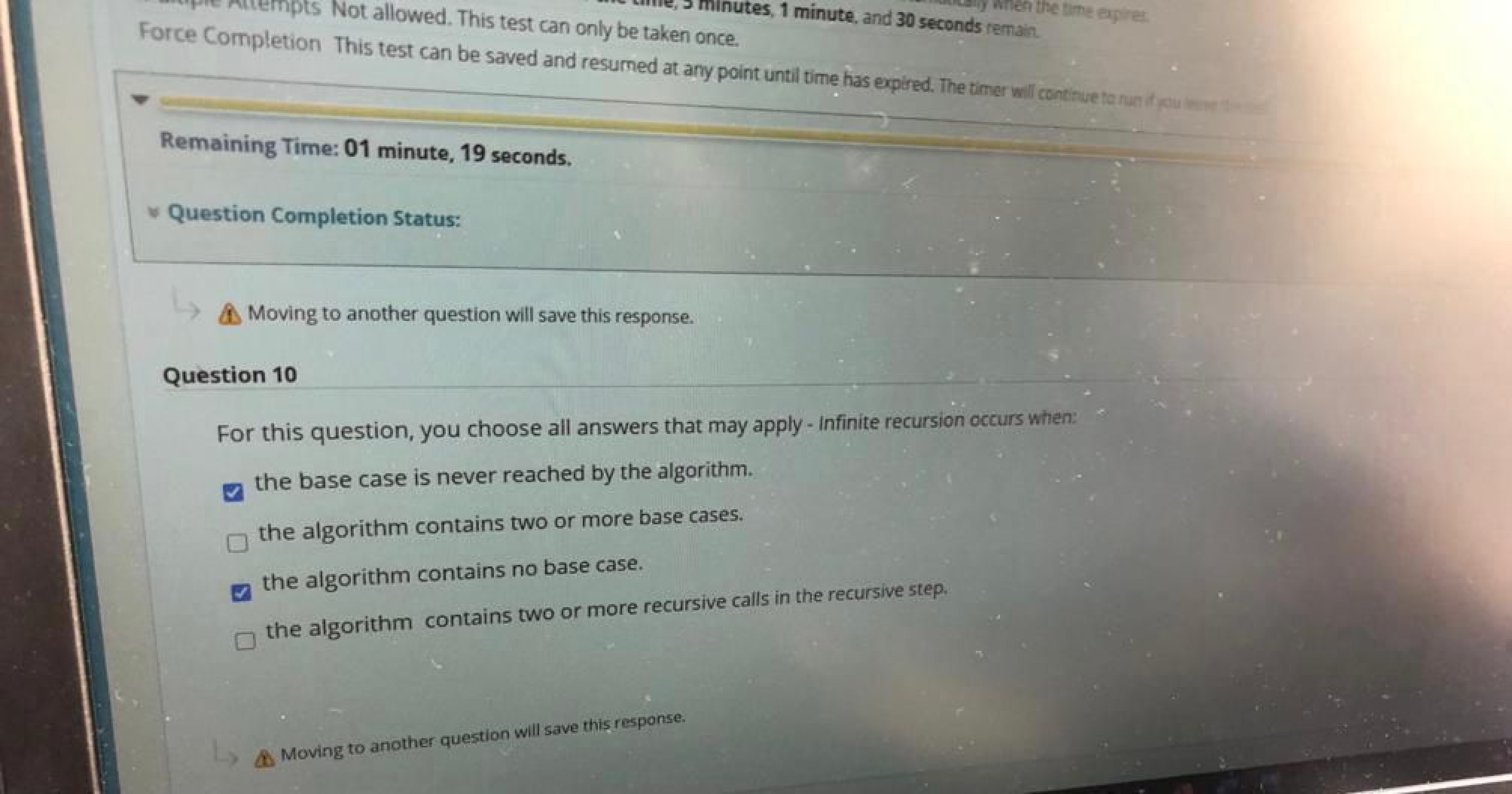
B) public int fact(int n){

int total=1;

for(int i=1;i<=n;i++)

total*=i;

return total; } // end method



…ically when the time expires.
…time, 5 minutes, 1 minute, and 30 seconds remain.

…ple Attempts Not allowed. This test can only be taken once.

Force Completion This test can be saved and resumed at any point until time has expired. The timer will continue to run if you leave the test.

Remaining Time: 01 minute, 19 seconds.

Question Completion Status:

Moving to another question will save this response.

Question 10

For this question, you choose all answers that may apply - Infinite recursion occurs when:

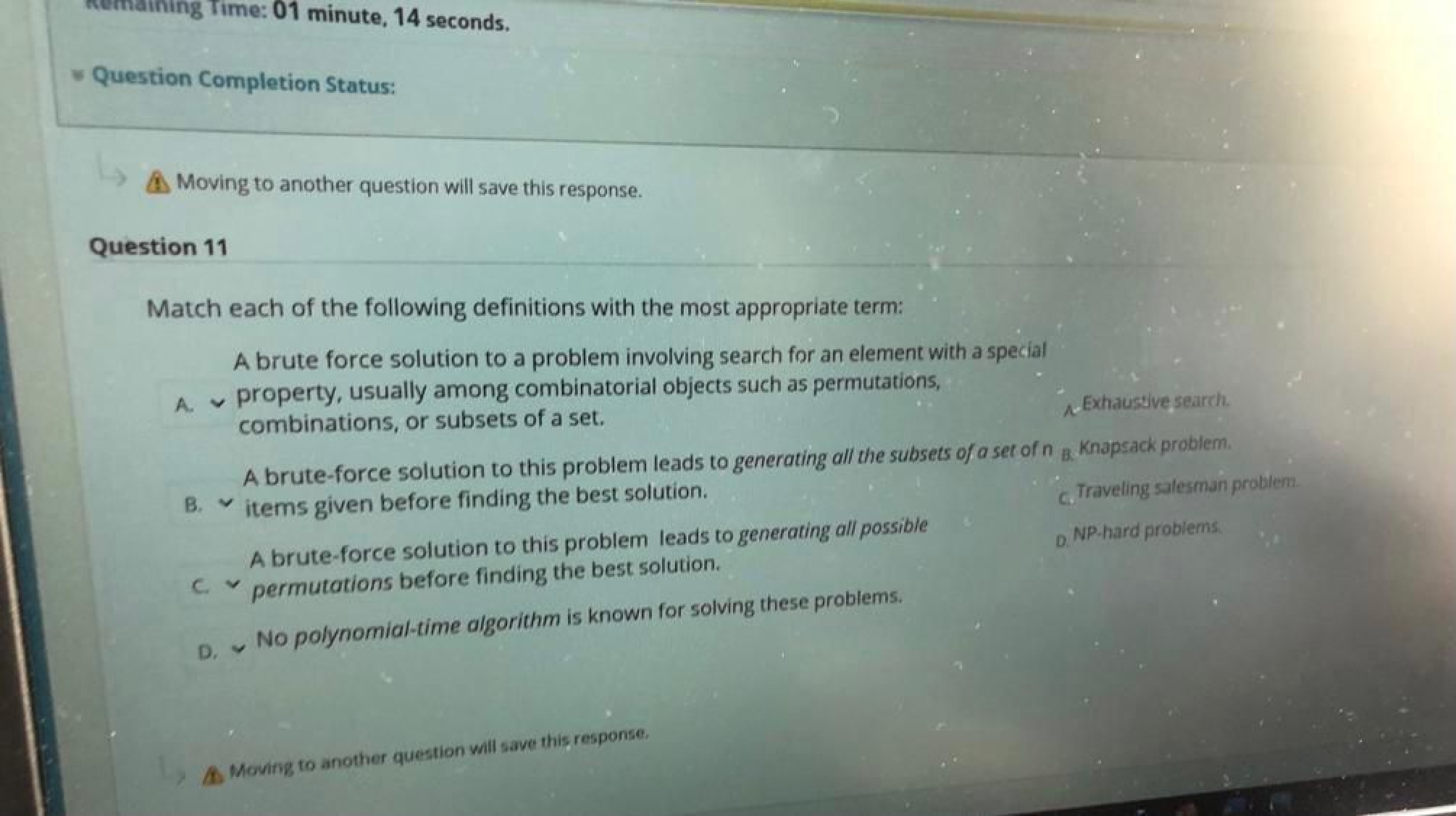
☑ the base case is never reached by the algorithm.

□ the algorithm contains two or more base cases.

☑ the algorithm contains no base case.

□ the algorithm contains two or more recursive calls in the recursive step.

Moving to another question will save this response.



Remaining Time: **01 minute, 14 seconds.**

▾ **Question Completion Status:**

⚠ Moving to another question will save this response.

Question 11

Match each of the following definitions with the most appropriate term:

A. ⌄ A brute force solution to a problem involving search for an element with a special property, usually among combinatorial objects such as permutations, combinations, or subsets of a set.

B. ⌄ A brute-force solution to this problem leads to *generating all the subsets of a set of n* items given before finding the best solution.

C. ⌄ A brute-force solution to this problem leads to *generating all possible permutations* before finding the best solution.

D. ⌄ No *polynomial-time algorithm* is known for solving these problems.

A. Exhaustive search.

B. Knapsack problem.

C. Traveling salesman problem.

D. NP-hard problems.

⚠ Moving to another question will save this response.

ng Time: 01 minute, 10 seconds.

▾ Question Completion Status:

Question 12

Master Theorem:

(case 1): If $a < b^d$, $T(n) \in \Theta(n^d)$

(case 2): If $a = b^d$, $T(n) \in \Theta(n^d \log n)$

(case 3): If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$

This is a matching questions: Match each of the following recurrences with its solution from the provided options, use the master theorem

E. ⌄ $T(n)=9T(n/3)+3n^2$

B. ⌄ $T(n)=3T(n/4)+6n$

F. ⌄ $T(n)=2^nT(n/2)+1$

D. ⌄ $T(n)=9T(n/3)+n$

A. $O(\log n)$

B. $O(n)$

C. $O(n\log n)$

D. $O(n^2)$

E. $O(n^2\log n)$

F. *Master method does not apply!*

Click **Submit** to complete this assessment.